# **Aste Online**

Nel seguente documento sono riportati gli artefatti prodotti nel corso del progetto.

**Sommario**

[Aste Online 1](#_Toc74155563)

[Diagramma dei casi d’uso 2](#_Toc74155564)

[Casi d’uso in formato dettagliato 3](#_Toc74155565)

[Chiude asta 3](#_Toc74155566)

[Attiva rilancio automatico 4](#_Toc74155567)

[Diagramma delle classi 5](#_Toc74155568)

[Design Pattern 6](#_Toc74155569)

[Asta\_Factory 6](#_Toc74155570)

[Offerte\_Listener 6](#_Toc74155571)

[Pagamento\_Adapter 6](#_Toc74155572)

[Vettore\_Adapter 6](#_Toc74155573)

[Diagramma degli stati 7](#_Toc74155574)

[Asta 7](#_Toc74155575)

[Diagrammi di sequenza 8](#_Toc74155576)

[chiudiAsta 8](#_Toc74155577)

[aggiungiOggetto 9](#_Toc74155578)

[aggiungiOfferta 9](#_Toc74155579)

[Contratti 10](#_Toc74155580)

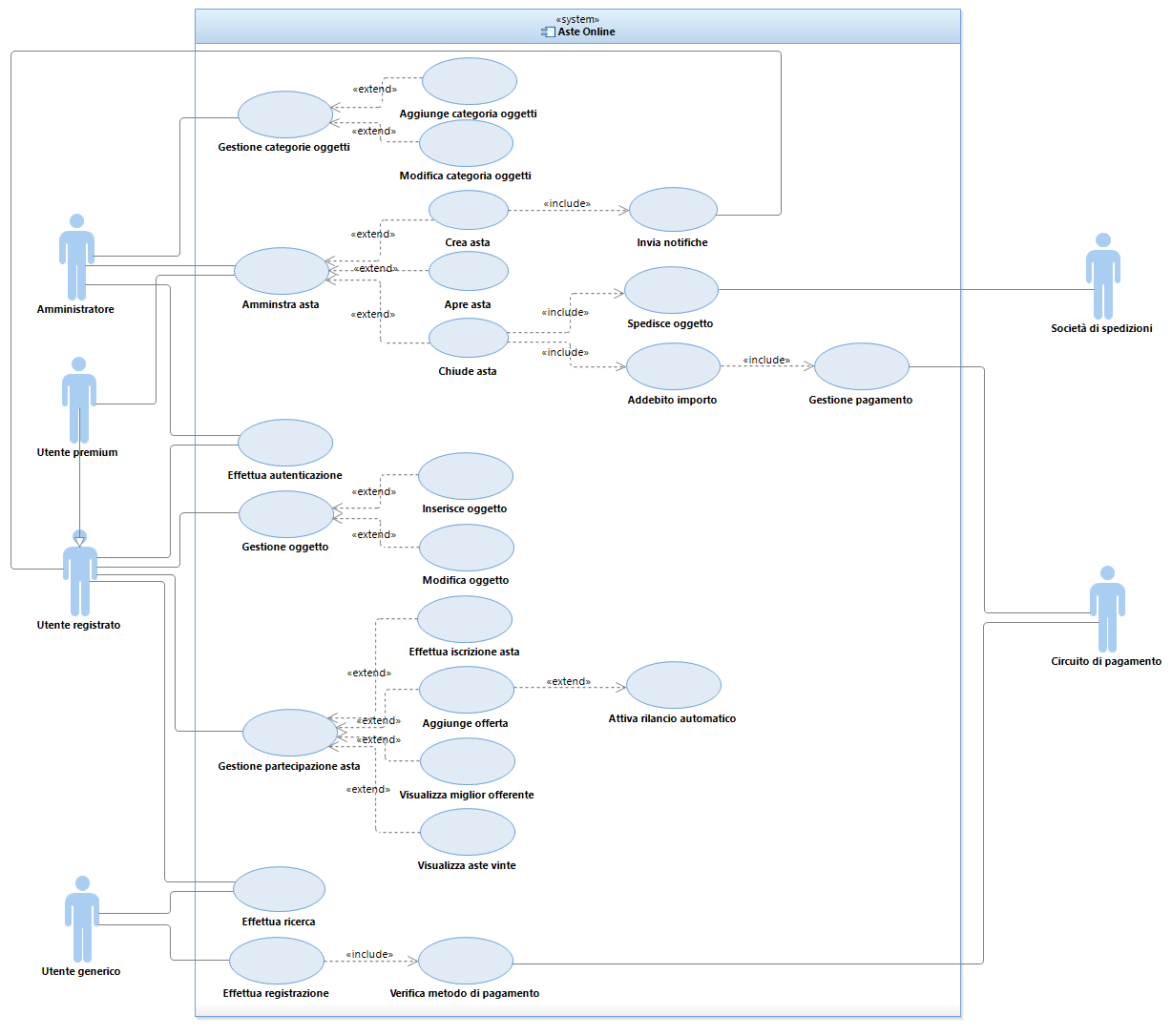
[aggiungiOfferta 10](#_Toc74155581)

[aggiungiOggetto 10](#_Toc74155582)

[Codice 11](#_Toc74155583)

[Asta 11](#_Toc74155584)

## **Diagramma dei casi d’uso**



## **Casi d’uso in formato dettagliato**

## **Chiude asta**

**Portata**: Aste Online

**Livello**: Obiettivo utente

**Attore primario**: Amministratore

**Attore finale**: Società di spedizioni, Circuito di pagamento

**Parti interessate ed interessi**:

* Amministratore: vuole concludere un’asta da lui gestita.
* Utente registrato: vuole ricevere l’oggetto che si è aggiudicato partecipando all’asta.
* Società di spedizioni: vuole ricevere le informazioni dell’oggetto da spedire (dimensione e peso) nel formato e nel protocollo corretto.
* Circuito di pagamento: vuole ricevere i dati del metodo di pagamento e l’importo da addebitare nel formato e nel protocollo corretto.

**Pre-condizioni**: L’amministratore è autenticato e la somma tra la data di inizio dell’asta e la durata dell’asta (espressa in ore) deve essere maggiore o uguale alla data e ora attuale.

**Garanzia di successo**: L’asta viene dichiarata chiusa; L’utente che ha fatto l’offerta più alta viene dichiarato vincitore; Un importo pari all’offerta vincitrice viene addebitato al metodo di pagamento inserito dall’utente vincitore; Viene avviato il processo di spedizione degli oggetti che erano stati messi all’asta.

**Scenario principale di successo**:

1. L’amministratore chiude l’asta;
2. Il sistema addebita l’importo all’utente;
3. Il circuito di pagamento conferma l’avvenuto pagamento;
4. Il sistema avvia il processo di spedizione per gli oggetti dell’asta;
5. La società di trasporto ritira e spedisce gli oggetti.

**Estensioni (o flussi alternativi)**:

1. Nessuno degli utenti partecipanti all’asta ha fatto offerte:
2. L’asta viene chiusa, ma non viene avviato nessun processo di addebito o di spedizione.
3. Il processo di addebito verso il vincitore dell’asta non viene concluso correttamente:
   1. L’utente viene escluso dall’asta e viene dichiarato vincitore l’utente che ha fatto la seconda offerta più alta.

**Frequenze di ripetizione**: In un giorno possono essere chiuse fino a 10 aste.

## **Attiva rilancio automatico**

**Portata**: Aste Online

**Livello**: Obiettivo utente

**Attore primario**: Utente registrato

**Parti interessate e interessi**:

* Utente registrato: vuole che il sistema, in seguito all’aggiunta di un’offerta da parte di un altro utente, faccia una nuova offerta a suo nome.

**Pre-condizioni**: L’utente è autenticato ed iscritto all’asta, l’asta è aperta.

**Garanzia di successo**: Il rilancio automatico viene attivato per l’utente che ne ha fatto richiesta, su una determinata asta e con le impostazioni inserite dall’utente.

**Scenario principale di successo**:

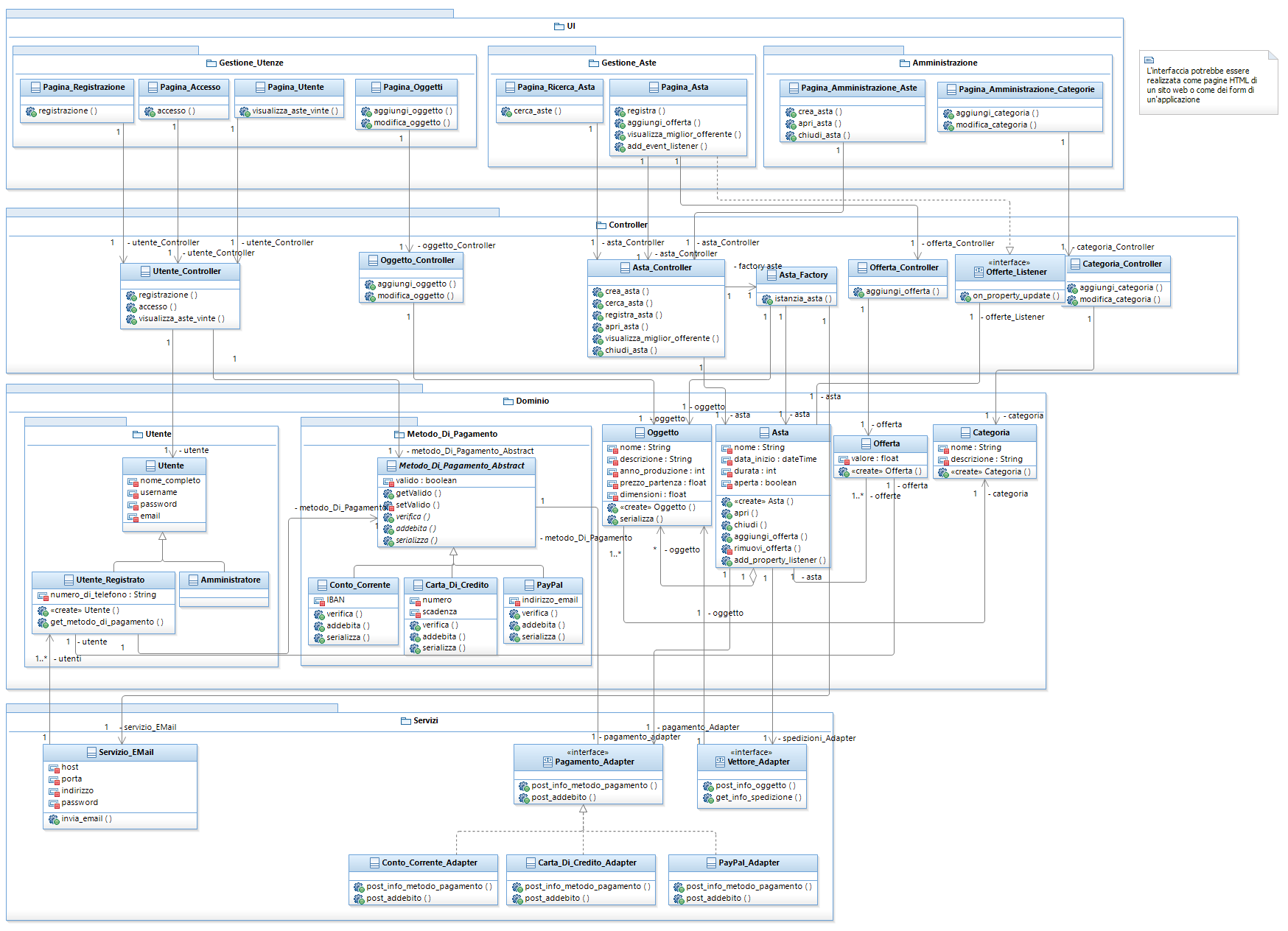
1. L’utente attiva il rilancio automatico, impostando l’importo ed il numero massimo di rilanci;
2. Il sistema, ogni volta che un altro utente effettuerà un’offerta sull’asta per cui è stato attivato il rilancio automatico e fintantoché il numero massimo di rilanci non viene raggiunto, genererà una nuova offerta per conto dell’utente;
3. Una volta che il numero massimo di rilanci è stato raggiunto, il sistema smetterà di rilanciare automaticamente.

**Estensioni (o flussi alternativi)**:

1. L’utente tenta di attivare il rilancio automatico dopo la chiusura dell’asta:
2. Il sistema mostrerà all’utente un messaggio di errore che indica che l’asta è terminata.

**Frequenza di ripetizione**: Ogni utente potrebbe attivare il rilancio automatico per ogni asta a cui è iscritto in qualunque momento, quindi quasi ininterrotto e dipendente dal numero di utenti registrati e di aste aperte.

## **Diagramma delle classi**



## **Design Pattern**

Nella progettazione delle seguenti classi sono stati applicati dei design pattern.

### **Asta\_Factory**

* + Design pattern: Factory;
  + Classi coinvolte: Asta\_Controller, Asta\_Factory, Asta, Oggetto, Servizio\_Email;
  + Descrizione: la factory si occuperà di istanziare l’asta assegnando il nome, data\_inzio, durata ed aggiungendo i vari oggetti, utilizzando le informazioni che il controller riceve dall’interfaccia. Inoltre, manda una email a tutti gli utenti registrati per notificare la creazione di un’asta.

### **Offerte\_Listener**

* + Design pattern: Observer;
  + Classi coinvolte: Pagina\_Asta, Offerte\_Listener, Asta;
  + Descrizione: Pagina\_Asta, che implementa l’interfaccia Offerte\_Listener, sarà “in ascolto” sulla proprietà offerte delle aste; ogni volta che un utente farà una nuova offerta, la pagina relativa a quell’asta si aggiornerà per mostrare la nuova offerta.

### **Pagamento\_Adapter**

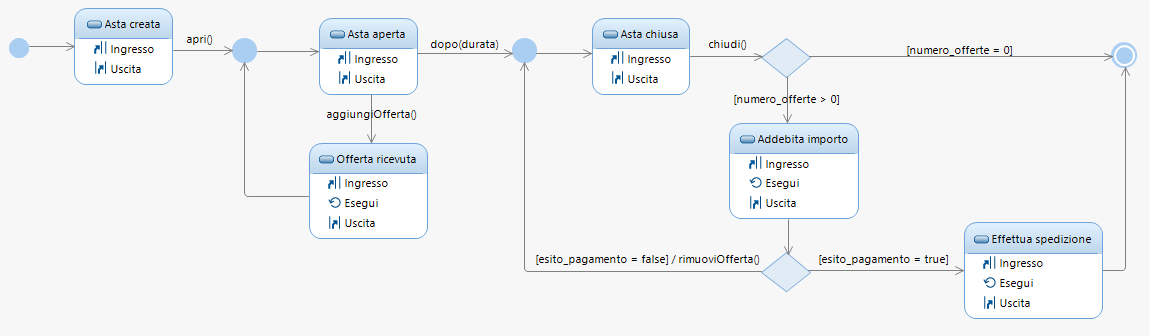
* + Design pattern: Adapter;
  + Classi coinvolte: Asta, Pagamento\_Adapter, Metodo\_Di\_Pagamento;
  + Descrizione: l’adapter invierà richieste all’endpoint specificato per verificare le informazioni del metodo di pagamento di un nuovo utente e per effettuare l’addebito quando un utente vince un’asta. Le specifiche della richiesta dipenderanno dal tipo di metodo di pagamento, e saranno definite nelle classi che implementeranno questa interfaccia.

### **Vettore\_Adapter**

* + Design pattern: Adapter;
  + Classi coinvolte: Asta, Vettore \_Adapter, Oggetto;
  + Descrizione: l’adapter, alla chiusura dell’asta e dopo che è stato effettuato l’addebito al vincitore, effettuerà una richiesta all’endpoint specificato per preparare la spedizione di un determinato oggetto; inoltre, dopo che è stata creata una spedizione, farà delle richieste per ottenere informazioni sulla spedizione. Le specifiche della richiesta dipenderanno dal vettore di spedizione scelto, e saranno definite nelle classi che implementeranno questa interfaccia.

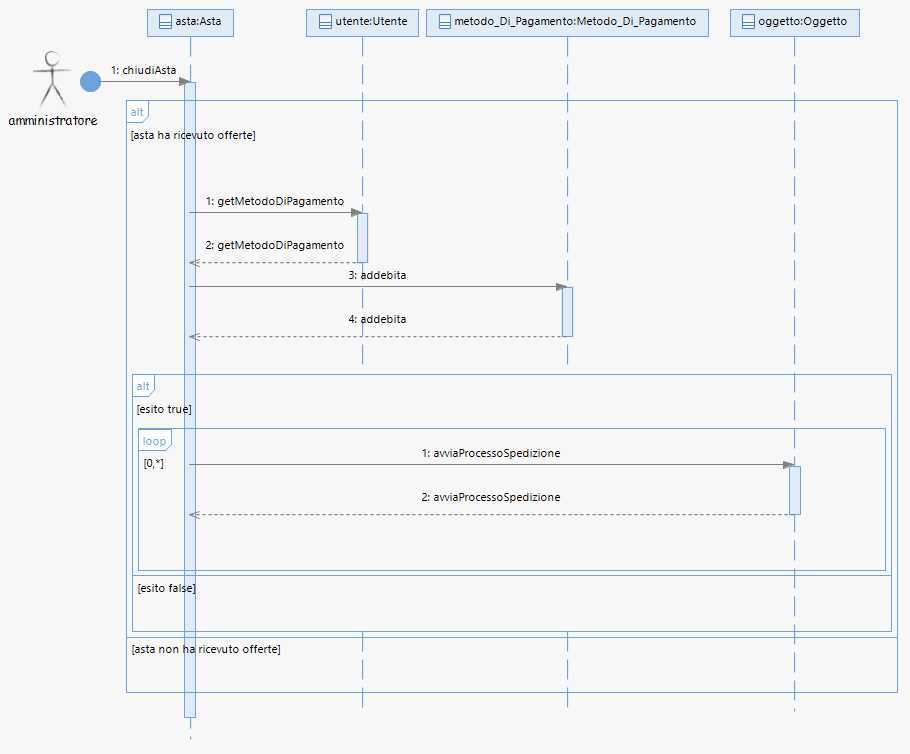
## **Diagramma degli stati**

### **Asta**

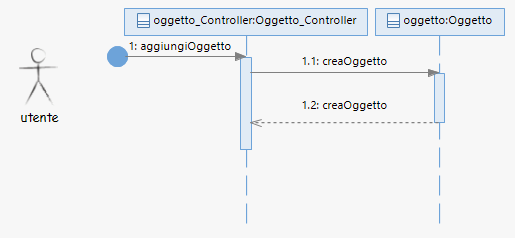


## **Diagrammi di sequenza**

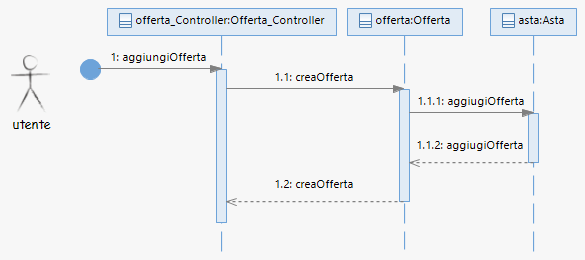
### **chiudiAsta**



### **aggiungiOggetto**



### **aggiungiOfferta**



## **Contratti**

### **aggiungiOfferta**

**Contratto: aggiungiOfferta**

**Operazione**: aggiungiOfferta(asta: Asta, valore: float)  
**Riferimenti**: Caso d’uso: Aggiunge offerta  
**Pre-condizioni**:

* L’utente è autenticato;
* L’asta è aperta (asta.aperta == true);
* L’utente partecipa all’asta (utente.asta contiene asta).

**Post-condizioni**:

* È stata creata un’istanza offerta di Offerta;
* Sono stati inizializzati i gli attributi di offerta con i parametri passati a aggiungiOfferta;
* offerta è stata aggiunta alla lista di offerte di asta (asta.offerta).

### **aggiungiOggetto**

**Contratto:** aggiungiOggetto

**Operazione**: aggiungiOggetto(nome: String, descrizione: String, anno\_produzione: int, prezzo\_partenza: float) **Riferimenti**: Caso d’uso: Inserisce oggetto  
**Pre-condizioni**: L’utente è autenticato  
**Post-condizioni**:

* È stata creata un’istanza oggetto di Oggetto;
* Sono stati inizializzati gli attributi di oggetto con i parametri passati a aggiungiOggetto;
* oggetto è stato aggiunto alla lista di oggetti dell’utente (utente.oggetto)

## **Codice**

### **Asta**

package aste\_online.dominio;  
  
import java.util.Date;  
import java.util.List;  
import java.util.TreeSet;  
  
/\*\*  
 \* Rappresentazione di un asta.  
 \*/  
public class Asta {  
  
 private String nome;  
 private Date data\_inizio;  
 private int durata;  
 private boolean aperta;  
 private List<Oggetto> oggetti;  
 private TreeSet<Offerta> offerte;  
  
 /\*\*  
 \* Crea una nuova asta.  
 \* **@param** nome Il nome dell'asta  
 \* **@param** data\_inizio La data in cui inizierà l'asta  
 \* **@param** durata La durata (in ore) dell'asta  
 \* **@param** oggetti La lista di oggetti da mettere all'asta  
 \*/  
 public Asta(String nome, Date data\_inizio, int durata, List<Oggetto> oggetti) {  
   
 this.nome = nome;  
 this.data\_inizio = data\_inizio;  
 this.durata = durata;  
 this.aperta = false;  
 this.oggetti = oggetti;  
 String messaggio = "E' stata creata una nuova asta: " + this.nome + ".\n"  
 + "Verra' aperta il " + data\_inizio.toString() + " e sarenno disponibili i seguenti oggetti:\n";  
  
  
  
 for (Oggetto oggetto : oggetti) {  
 messaggio += oggetto.getNome() + "\n";  
 }  
 // Recupera gli utenti dal DB  
 // List<Utente> utenti = ...  
 Servizio\_EMail.invia\_email(messaggio, utenti);  
 }  
  
 /\*\*  
 \* Apre l'asta, in modo che gli utenti possano effettuare delle offerte.  
 \*/  
 public void apri() {  
  
 aperta = true;  
 }  
  
 /\*\*  
 \* Chiude l'asta, fa partire il processo di pagamento al vincitore ed il processo di spedizione per gli oggetti.  
 \* Se non è stata effettuata nessuna offerta, l'asta viene semplicemente chiusa.  
 \*/  
 public void chiudi() {  
  
 aperta = false;  
 if (offerte.size() > 0) {  
 boolean esito = Pagamento\_Adapter.post\_addebito(offerta.utente.getMetodoDiPagamento.serializza(),  
 offerte.first().getValore());  
 if (esito) {  
 for (Oggetto oggetto : oggetti) {  
 String risultato = Vettore\_Adapter.post\_info\_oggetto(oggetto.serializza());  
 // Salva il risultato...  
 }  
 } else {  
 // Se il pagamento non va a buon fine, rimuovo l'offerta vincitrice e riprovo con la seconda offerta più alta.  
 rimuoviOfferta();  
 chiudi();  
 }  
 }  
 }  
  
 /\*\*  
 \* Aggiunge un'offerta alla lista ordinata.  
 \* **@param** offerta La nuova offerta  
 \*/  
 public void aggiungiOfferta(Offerta offerta) {  
  
 offerte.add(offerta);  
 }  
  
 /\*\*  
 \* Rimuove l'offerta più alta. Da usare quando l'addebito al vincitore non va a buon fine per far proseguire il  
 \* processo di chiusura.  
 \*/  
 private void rimuoviOfferta() {  
  
 offerte.remove(offerte.first);  
 }  
  
 /\*\*  
 \* Aggiunge un listener sulla proprietà offerte; la finestra dell'asta verrà aggiornata ogni volta che un utente  
 \* fa una nuova offerta.  
 \*/  
 public void addPropertyListener() {  
  
 // ...  
 }  
}